2016

Convolutional neural networks as feature extractors for data-scarce visual searches

ben Abdallah, Hichem

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/50597

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CONVOLUTIONAL NEURAL NETWORKS AS FEATURE EXTRACTORS FOR DATA-SCARCE VISUAL SEARCHES

by

Hichem ben Abdallah

September 2016

Thesis Advisor:	Mathias Kolsch
Second Reader:	Magdi Kamel

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2016	Master's Thesis 01-29-2016 to 09-02-2016

4. TITLE AND SUBTITLE

CONVOLUTIONAL NEURAL NETWORKS AS FEATURE EXTRACTORS FOR DATA-SCARCE VISUAL SEARCHES

5. FUNDING NUMBERS

6. AUTHOR(S)

Hichem ben Abdallah

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Image classification is one of the core problems in Computer Vision. The classification task consists of predicting a single label from a fixed set of categories for a single image. To perform an image classification, the classifier should consider the semantic identity of the image rather than irrelevant characteristics and variations such as the coincidental contrast or brightness of the images or the type of background. Applying Convolutional Neural Networks (CNNs) as feature extractors is a powerful approach to image classification. Training these CNNs necessitates a tremendous amount of training samples, and it is costly in terms of computational time. Since it is not guaranteed that one can find a sufficient amount of training data for a specific class target, we are conducting transfer learning of a CNN model (learned from a large data set) to generate a new representation of the images. These representations are classified with K-Nearest Neighbors within a target space that has just a few training samples. We aim to define the appropriate parameters including distance metric, layer from which to extract features, and minimum number of training samples to be considered to obtain the best classification results with our approach.

14. SUBJECT TERMS

Convolutional Neural Networks, k-Nearest Neighbors, image classification, data scarcity, transfer learning, activation codes, high-dimensional space, cosine similarity, Euclidean distance, t-SNE

15. NUMBER OF PAGES 83

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

CONVOLUTIONAL NEURAL NETWORKS AS FEATURE EXTRACTORS FOR DATA-SCARCE VISUAL SEARCHES

Hichem ben Abdallah
Captain, Tunisian Army
M.S., Tunisia Polytechnic School, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2016

Approved by: Mathias Kolsch
Thesis Advisor

Magdi Kamel
Second Reader

Dan C. Boger
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Image classification is one of the core problems in Computer Vision. The classification task consists of predicting a single label from a fixed set of categories for a single image. To perform an image classification, the classifier should consider the semantic identity of the image rather than irrelevant characteristics and variations such as the coincidental contrast or brightness of the images or the type of background. Applying Convolutional Neural Networks (CNNs) as feature extractors is a powerful approach to image classification. Training these CNNs necessitates a tremendous amount of training samples, and it is costly in terms of computational time. Since it is not guaranteed that one can find a sufficient amount of training data for a specific class target, we are conducting transfer learning of a CNN model (learned from a large data set) to generate a new representation of the images. These representations are classified with K-Nearest Neighbors within a target space that has just a few training samples. We aim to define the appropriate parameters including distance metric, layer from which to extract features, and minimum number of training samples to be considered to obtain the best classification results with our approach.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Acronyms and Abbreviations

ANN	Artificial Neural Network
BVLC	Berkeley Vision and Learning Center
CNN	Convolutional Neural Network
CONV	Convolutional
CS	Cosine Similarity
DOD	Department of Defense
ED	Euclidean Distance
FC	Fully Connected
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Competition
k**-NN**	K-Nearest Neighbor
NN	Neural Network
NPS	Naval Postgraduate School
PCA	Principal Component Analysis
ReLU	Rectified Linear Units
SVM	Support Vector Machine
TP	True Positive
t-SNE	t-distributed Stochastic Neighbor Embedding

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

First of all, all praise and thanks are due to Almighty God for providing me with the understanding, insight, and passion to carry out this master's research. I would like to express my deepest gratitude to my advisor, Dr. Mathias Kolsch, who never stopped challenging and helping me throughout this journey. I wish to express my warm appreciation to Dr. Magdi Kamel for his intellectual comments and recommendations. I would like to acknowledge with many thanks Tom Batcha for his generous support worthy of recognition. I take this opportunity to express my sincere thanks to Academic Associate Glenn Cook for his guidance and help. I am grateful to my family for their unceasing encouragement and love. Finally, I thank everyone who directly or indirectly supported me to accomplish this present work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

Image classification is one of the core problems in Computer Vision. The classification task consists of predicting a single label from a fixed set of categories for a given pixel matrix that represents a single image. In machine learning, to classify images into different categories, a data-driven approach is used. Instead of developing an algorithm that provides coded definitions of the categories of interest, training examples are provided to the machine, which learns a model representation of each class from the visual appearance.

Data driven learning can be supervised or unsupervised. In the case of supervised learning all the samples must be labeled; however, in the case of unsupervised learning, the samples are not labeled and learning algorithms generally try to cluster the data into different categories by their similar features. For the supervised learning, the process of image classification consists of input, learning, and evaluation. The input dataset, called the training set, is a set of images labeled with one of the different output classes. The learning phase consists of training a model that will be used to classify images. The evaluation task refers to the quality assessment of the prediction of labels for a new set of examples, called the test set. A good classification model results in high matching between the predictions and the ground truth. To perform an image classification, the model should consider the semantic identity of the image rather than irrelevant characteristics and variations such as the coincidental contrast or brightness of the images or the type of background.

Recent improvements in infrastructure and hardware computation have enabled computer vision researchers to enhance the techniques they are using to fulfill several tasks like classification, object recognition, and tracking. The definition of robust image descriptors like SIFT [1] and ORB [2] has allowed the move from controlled to arbitrary setups. On the other hand, computer vision researchers have been moving more towards using Convolutional Neural Networks (CNNs) to extract images' features. Several studies demonstrate the power of the generic descriptors extracted from CNN [3], [4].

Neural networks are typically arranged in multiple layers of neurons [5]. An image vector is received by the very first layer and then transformed through a sequence of hidden layers before reaching the output layer. At each hidden layer, there are completely independent neurons that are connected to all neurons of the previous layer. Therefore, to take advantage of the image nature of the input, CNNs provide a more suitable model architecture. In fact, the neurons at each layer of a CNN are arranged in three dimensions: width, height and depth [5]. For an image input, the depth represents the three color filters (Red, Green, Blue). Moreover, the neurons in a layer N will only be connected to specific neurons at nearby locations rather than being connected to all neurons in the layer N-1.

Technically, the CNN approach has three elements: (1) a score function that assigns a calculated score to the input data, (2) a loss function that measures the consistency between the ground truth and the predicted labels, and (3) an optimization process that minimizes the loss function by determining the adequate set of model parameters [5]. Once the learning is complete the training set is discarded and only the learned parameters are kept.

In practice, training a CNN from scratch is uncommon because of the scarcity of a dataset with sufficient size. As an alternative, one can start with a CNN pretrained on a huge dataset for parameter initialization and feature extraction. The practice of adapting this pretrained model to a new, distinct set of classes is called *Transfer Learning*.

The general idea is to use large and deep neural networks to learn powerful feature extractors for images of any content. Once learned, these feature extractors enable the generation of new representations of the input images, which are utilized to perform the objects' recognition and classification tasks. CNNs are trained to perform supervised learning [4] as well as unsupervised learning [6], [7].

The problem is that training a deep neural network model is expensive in terms of computation resources and time and requires large amounts of data. In addition, new data are expensive in terms of labeling costs and time. The cost issue highly restricts the amount of data, which in turn impacts the accuracy of the classification task. Our goal is to leverage the scarce training data in order to achieve good classification results.

Our approach to solve the stated problem is to make maximum use of the old data since it is common within the computer vision community to share pretrained models so that others can benefit from the released CNN weights. For example, the Berkeley Vision and Learning Center (BVLC) released the Caffe deep learning framework [8], and shared Caffe trained models for unrestricted use. Additionally, many huge image datasets with object annotation such as ImageNet [9], which contains 1.2 million images for 1,000 classes, and Pascal VOC [10] are publicly released and can help train and test CNNs.

The purpose of the present research study is to conduct transfer learning of a deep CNN model learned from a large data set and reutilize this knowledge in training a classifier for the new data that has just a few training examples. Throughout the present thesis work, we look for answers for the following research questions:

1. How effective is the transfer learning approach to classify features of different sets of images?
2. What would be the best layer at which to extract the CNN codes?
3. Which metric achieves better classification performance in a high dimensional space?
4. What is the effect of size of the training set on the classification results?

Our idea is to conduct transfer learning of deep CNN models and reutilize the former knowledge in training a K-Nearest Neighbor (k-NN) classifier for the new data that has only a few training samples. This thesis attempts to recommend a transfer learning method to optimally utilize a few data samples in order to make accurate image classifications. The present research work aims to determine which configuration is more likely to tolerate the data scarcity while providing more accurate classification.

The advantage of this approach enables the Department of Defense (DOD) branches to develop sophisticated classification systems to accurately identify new weapons, aircrafts, or submarines that currently have few images. The shortage of images to train the systems hinders the ability to identify such military equipment, but with our new approach, only a few images are needed. This is beneficial in reducing friendly fire incidents and supporting tactical decision making.

The thesis research is organized as follows. The first chapter gives a general introduction to our work. The second chapter introduces the CNNs in terms of motivation

and architectures, and gives an overview of the extraction and transfer of features learned from pretrained CNNs. We present some related works with respect to CNNs and transfer learning. The third chapter presents our methodology to solve the classification problem in the presence of scarce data. The fourth chapter provides and discusses the results of our solution. The fifth chapter concludes our work and suggests avenues for future work. Appendix A describes the main algorithm we use to apply our method. Appendix B shows more outputs of our experiments in terms of number of True Positive (TP) and False Positive (FP) instead of F-scores presented in the chapter four. Appendix C presents the results of our experiments by target classes.

CHAPTER 2:
Background

2.1 Foundations

2.1.1 Convolutional Neural Networks

CNNs, introduced by LeCun et al. [11], are Artificial Neural Networks (ANNs) that have specific topology and specific non-linear functions. In fact, the term “Convolutional (CONV)” indicates that CNNs use the convolution mathematical linear operation in at least one of their layers [12]. CNN architectures hold a large number of parameters that are learned while training the model on the training dataset [13].

2.1.1.1 Biological Motivation

Biological neural networks gave rise to ANNs that replicate the most fundamental functions of the central nervous systems. A neuron is a basic computational unit of the central nervous system. At each neuron, dendrites receive input signals and axons carry the produced output signals [5]. Through the synapses, neurons connect to dendrites of another neuron, which allows the transmission and the summation of signals. Figure 2.1 demonstrates a common mathematical model of a neuron.

Li and Karpathy [5] describe the computational model of a neuron, in which the signals that travel along the axon interact multiplicatively with the next neuron’s dendrites, based on synapse strength. They further explain that the “synaptic strengths (the weights W) are learnable and control the strength of the influence” of one neuron on another. Once the signals are summed, the neuron fires a spike along its axon only “if the final sum is above a certain threshold” [5]. The neuron’s firing rate is modeled with an activation function (f) representing the frequency of the spikes along the axon. The authors indicate that the Sigmoid function σ is a common activation function choice, which takes the sum of the signals’ strength and normalizes it to a range between 0 and 1.

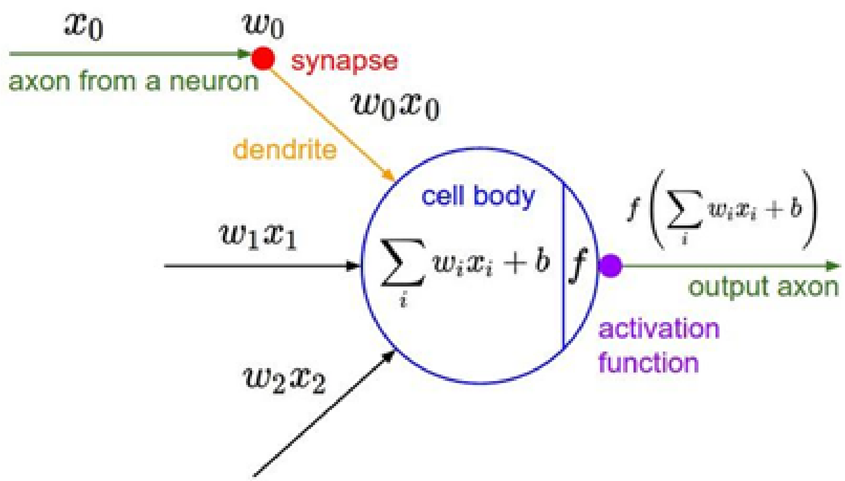

Figure 2.1. Mathematical Model of a Neuron.

Source: [5, Figure 1].

2.1.1.2 Neural Network Architectures

2.1.1.2.1 Layer-wise Organization

Neural Networks (NNs) are organized in a layer-wise manner: an input layer, one or multiple hidden layers, and an output layer [5]. NNs are modeled as graphs where the output of some vertices (neurons here) are the inputs of others. The graph that represents an NN must be a finite directed graph without cycles known as an acyclic graph [5]. In fact, having cycles in the NN leads to infinite loops in both forward and backward propagation.

2.1.1.2.2 Neural Network Topologies

Commonly, NN graphs are represented by layers of neurons. Each neuron is connected pairwise to neurons from adjacent layers yet does not share any connections with neurons from the same layer. This topology is called *Fully Connected (FC)* layers, and it is shown in Figure 2.2.

A CNN model consists of several combinations of CONV and FC layers, with application of a nonlinearity function at the end of each layer. A common architecture of a CNN model may consist of many stages of layers, and each stage consists of convolution, nonlinear activation, and optionally Pooling layers. Convolutional layers produce a set of linear activations by performing several parallel convolution operations, and then nonlinear

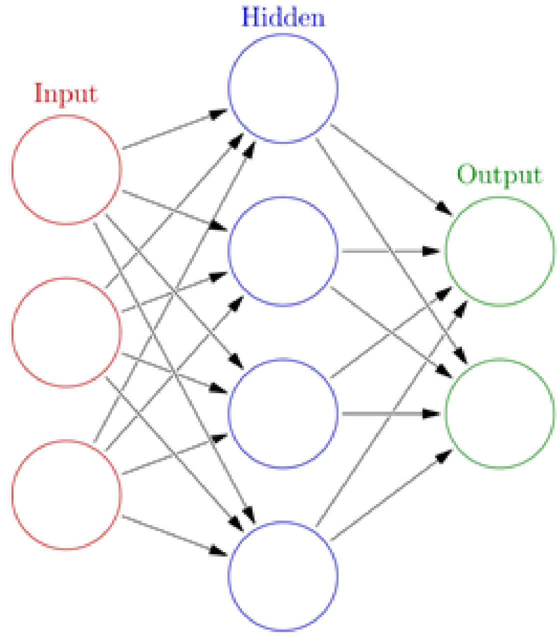

Figure 2.2. Fully-Connected Layers.

Source: [5, Figure 2].

activation functions are applied to each of the linear activations. As a result, a nonlinear decision boundary is produced via nonlinear combinations of the weighted inputs. The most common nonlinear activation functions are the Sigmoid and Rectified Linear Units (ReLU) functions. An optional non-linear down-sampling may be applied through a Pooling function. According to Li and Karpathy [5], it is common to apply a Pooling function in between two CONV layers in order to decrease the spatial size of the representation, which in turn leads to a decrease in the number of parameters and computation load in the network. The most common Pooling functions are max Pooling, average Pooling, and L2-norm Pooling.

2.1.1.3 Features

Instead of using hand-coded feature extractors and descriptors, recent researches use CNN models to extract images' features. According to Fisher et al. [14],"CNNs clearly outperform SIFT [1] on descriptor matching." Specific locations in the images such as structures' peaks or corners represent an important category of features. The description of these localized

features, called “key points” or “interest points,” are patches of pixels surrounding the point locations. Important feature categories also include edge profiles, such as the silhouette of building roofs. These features match based on their orientation and local appearance [15].

According to Yosinski et al. [16], features that look like color blobs or Gabor filters are learned by the first layers of modern deep NNs when trained on images. These first-layer features are called “general” as these standard features can be extracted independently from the task and the dataset. However, last-layer features depend significantly on the task and dataset. For instance, training a neural network for a supervised classification task with X target classes implies an assignment of an output unit to each of the X neurons at the output layer [16]. Therefore, these last-layer features are called “specific.” The learned features within a network are eligible for repurposing or being transferred to a another network if they are such “general” features [17], [18].

2.1.2 Transfer Learning

2.1.2.1 Overview

Transfer learning is an emergent learning framework based on reusing common knowledge extracted from existing systems. In some cases, training data might only be abudant in domains that differ from our classification task domain; or, the abundant data may follow a different data distribution. “In such cases, knowledge transfer, if done successfully, would greatly improve the performance of learning by avoiding much expensive data-labeling efforts” [19] and reducing the time of building models from scratch.

2.1.2.2 Transfer Learning Scenarios

Since training a CNN on ImageNet [9] can be very time consuming, even on Graphics Processing Unit (GPU) computing enabled hardwares, computer vision researchers release their final CNN parameters for the benefit of others who can reuse the networks and apply the transfer learning approach. Two major transfer learning scenarios [16] can be used:

1. **Fine-tuning**: having a pretrained CNN, retrain the entire network to a different task. This end-to-end feedforward and backpropagation will allow the adjustment of the learned weights with regards to the novel task. There are two possibles options: we

can either fine-tune all the layers of the CNN or fix the first layers and only fine tune the last portion of the CNN [5].

2. **Freezing the transferred layers**: having a CNN pretrained on a large number of samples, remove the last fully-connected layer, then replace it by a new classifier (e.g., Linear Support Vector Machine (SVM) or Softmax classifier) to be trained in the novel task on top of the CNN on the new dataset.

2.2 Previous Works

Contemporary research results demonstrate the power of the generic representations or descriptors extracted from CNN in various visual tasks.

Donahue et al. [20] have evaluated the performance of linear classifiers, such as SVM, on features extracted from a deep convolutional neural network trained in the fully supervised setting. The authors demonstrated that features learned from a CNN trained with an auxiliary large labeled dataset possess enough representational strength and generalization ability to efficiently achieve semantic visual recognition. Their approach consists of extracting various activations from different hidden layers of the deep CNN as features, then analyzing linear classifier performance. Even though the difference between tasks is great, initializing a CNN with transferred features from unsimilar tasks performs better than resetting with random weights [16].

The work of Razavian et al. [21] adds to the mounting evidence that the generic descriptors extracted from a CNN are very powerful. Their work differs from Donahue et al. [20] by the use of a different framework, the OverFeat convolutional neural network [22], which was trained in order to perform object classification in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2013. Yet, the 4096 dimensional features extracted from this CNN as generic representations are input to the SVM classifier to solve different classification tasks applied to diverse sets of data. A one-against-all strategy is applied whenever the labels are not mutually exclusive; otherwise, one-against-one linear SVMs with voting are applied. These representations are extracted from the first fully connected layer (layer 22) of the network and input to a linear SVM classifier or an L2 distance in case of instance retrieval. Their consistent results strongly recommend the features extracted from deep learning of CNNs as primary candidates in most recognition

tasks in the computer vision field.

Our present thesis work is part of mounting evidence [20], [22]–[25] that CNNs offer a means to learn generic descriptors transferable to several visual recognition and classification tasks. In addition, our approach represents an extension of Donahue et al.'s work [20] and Razavian et al.'s work [21] in the way that we still extract features learned from a CNN trained with a large labeled dataset. Yet, we apply a different classifier, the k-NN technique, and we investigate the best metric for assessing the similarity among the new representations in a high dimensional space.

Li et al. [13] relied on the generic descriptors extracted from CNNs to discover mid-level visual elements within a given dataset. They applied a data mining approach known as pattern mining through association rule mining to find conforming patterns between new images' representations extracted from CNNs.

The following chapter presents our methodology to solve the problem of scarcity of training samples while applying the transfer technique to take advantage of the powerful features extracted from a pretrained CNN with an abundant training.

CHAPTER 3:
Methodology

Throughout this chapter, we describe our approach to accurately classify out-of-sample images in the presence of few training data samples. We discuss how to transfer learned knowledge from a CNN model trained on a large dataset to train our classifier. Our method aims to determine the best parameters that better tolerate the data scarcity. Then, we present the dataset to which we apply our method during the experiment phase of our thesis.

3.1 Phase 1: Transfer Learning

In our method, we consider transferring parameters learned from a CNN model trained on a large dataset instead of training a new model from scratch. We adopt one of the publicaly released network settings; then we feed the training samples to our network in order to generate new representations of these images. We are not interested in learning new features from these training samples since their number is restricted. Thus, we opt for freezing the transferred layers by setting the learning rate at each layer of the network at zero. Since we choose to work with the Caffe framework [8] trained on the ImageNet dataset, our network consists of seven layers. The first five layers are CONV layers while the two last layers are FC layers. We consider extracting the new representations, called activations, at each layer.

3.2 Phase 2: Data Visualization with t-Distributed Stochastic Neighbor Embedding

The next step in our method is to visualize the activations extracted at each layer. At this phase, we are interrested in the spatial distribution of the activations. Since our new representation is of high dimension, we apply the t-distributed Stochastic Neighbor Embedding (t-SNE) [26] technique for dimensionality reduction to visualize our data. The output of this technique consists of a low dimension embedding. We then plot the mapped low dimensional data. From visual observations, we decide which layers should be considered for further search. We are looking for how discriminative the representations

are at each layer.

3.3 Phase 3: K-Nearest Neighbors Classifier

The activations extracted from our network at the layers considered in the previous phase are fed to a k-NN classifier. k-NN is an instance-based learning algorithm that stores all training samples. Given an out-of-sample test instance, it locates a number k of closest training samples by means of a distance function or similarity metric. If a majority of the k training samples have the same label, this label is the prediction for the test sample. In case of a tie, no prediction is made. In this thesis, we take advantage of the simplicity and effectiveness of the k-NN method to classify a new image instance relying on its CNN representation.

Furthermore, we investigate two main concerns:

1. Which metric achieves better classification performance in a high dimensional space?
2. What is the impact of the number of training instances per class with respect to the accuracy of the k-NN classification?

Answering the first question, we compare the Cosine Similarity (CS) and Euclidean Distance (ED) to determine the better metric for our high dimensional feature space. In our approach, given a training set, an out-of-sample test instance, and the parameter k, we look for the nearest neighbors for both metrics. From this list of nearest neighbors for each metric, we predict the most likely class for the test instance with the k-NN classifier. We compare the classification accuracy for both metrics at different layers, for different sizes of training sets, and different numbers of nearest neighbors. Thus, we evaluate the hyperparameter $k \in \{1,3,5\}$ to compare the output of k-NN for different values of k. The classification accuracy is calculated from comparing the predicted class with the expected class.

The impact of the number s of training instances per class is addressed by evaluating TP and FP as output of the classification task while varying the hyperparameter s. The straightforward implementation of the design-of-experiments is described in Appendix A. The results of our implementation tested on the logos dataset are shown and discussed in the following chapter.

3.4 The Logos Dataset

Our training dataset is retrieved from the logos dataset available online at [27], [28]. The logos dataset is composed of 10,000 images covering several domains such as sports, culture, and personalities. The format of all images is JPEG whether they contain a logo or not. Images are resized to 800 by 800 pixels. Figure 3.1 shows a sample from the logos dataset.

Figure 3.1. An Example of Logos Data.
Source: [27], [28].

Additional metadata for each image contains an image name, boolean that indicates whether the image contains a logo, and the coordinates of the logo bounding-box within this image if applicable. The location of the logo is not provided for all images.

As the number of samples differs from one class to another, from 36 ground truth classes we keep only 19 classes having at least 50 samples each. The first step is to prepare the data by cropping the training images based on the bounding-box coordinates provided with the dataset. Figure 3.2 shows samples of cropped logos to be fed to the CNN to generate new representations or activation codes at the layers FC6 and FC7.

Figure 3.2. Samples of Cropped Logos Data.
Adapted from [27], [28].

The logos data is split into a training set and one test sample to conduct a one-against-all classification task. As we do a k-fold cross-validation, with 50 samples per class, we do 50 one-against-all training and classification steps. We define the training set in an iterative loop in such a way that we have one more sample from each class at every iteration. This helps us determine the impact of the size of the training set and the minimun required number of training samples to achieve a desired classification accuracy. The following chapter provides and discusses the results we obtain by applying our described method to the logos dataset.

CHAPTER 4:
Experiments and Results

In this chapter, we investigate the spatial arrangement of the high-dimensional features extracted from different layers of our CNN by projecting them into a 2-D space. Once we determine at which layers the features are mmore easily separable, we conduct further experiments based on the k-NN method presented in the previous chapter. We strive to determine the most adequate parameters, including number of nearest neighbors, distance metric, layer from which the features will be extracted, and the number of training samples that best handle the problem of data scarcity while providinig good classification results.

4.1 Experiments Environment

We run our experiments within the following environment:

1. Caffe framework [8],
2. GoogleNet pretrained model provided with Caffe,
3. GPU node at the High Performance Computer located at the Naval Postgraduate School (NPS),
4. Personal laptop: Intel Core(TM) i7 2.2GHz processor and 4GB memory,
5. Platform: Anaconda 3, Jupyter Notebook,
6. Programming language: Python 3.

4.2 Spatial Arrangement of the Features

The activations extracted from the pretrained CNN are saved by classes and by layers in different *csv* files. Entries of each file are high dimensional arrays. We choose randomly four logo classes: Adidas, Puma, Ferrari, and Mercedes to visualize their activations. We feed the activations of these four classes at each layer to the t-SNE algorithm [26]. We import the t-SNE algorithm from the *sklearn.manifold* Python library. Our model is created as follows:

```
model = TSNE(n_components=2, random_state=0, init='pca',perplexity=30)
```

This generates a 2-dimensional embedding of the high dimensional space at each layer. The *pca* parameter of the model indicates that we are reducing the number of dimensions by the Principal Component Analysis (PCA) dimensionality reduction method before applying t-SNE technique. This helps speed up the pairwise distances computation between training instances [26]. Then, we plot the data points at the low-dimensional embedding as colored points based on their semantic classes.

We extract and analyze activation codes from different layers of our CNN for the different classes. The parameters of our CNN are learned and transferred from a CNN trained on the ImageNet dataset. We then feed our target classes to the network and extract the activation codes at each layer. During our experiments, we freeze the model parameters as a transfer learning option by setting the learning rate to zero for each layer.

Figures 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, and 4.7 show the spatial arrangment of the activations representing every input image extracted from the layers of the pretrained CNN.

Our main concern is to compare the dispersion of the features in 2-D space to determine at which layer these representations are more compact (low intra-cluster distances) and discriminative (high inter-cluster distances). The results show that at early layers the features are less discriminative and more disperse. As we move towards the last layers, the features show less dispersion while the separate classes are well discernible. Note that t-SNE minimizes the Kullback-Leibler divergence between the distribution in the high dimension space (that measures pairewise similarities of the input samples) and the distribution in the low-dimensional embedding (that measures pairwise similarities of the corresponding points) [26]. Thus, we consider the dispersion of the features evaluated at the 2-dimensional space to reflect the dispersion of the 4096-dimensional space.

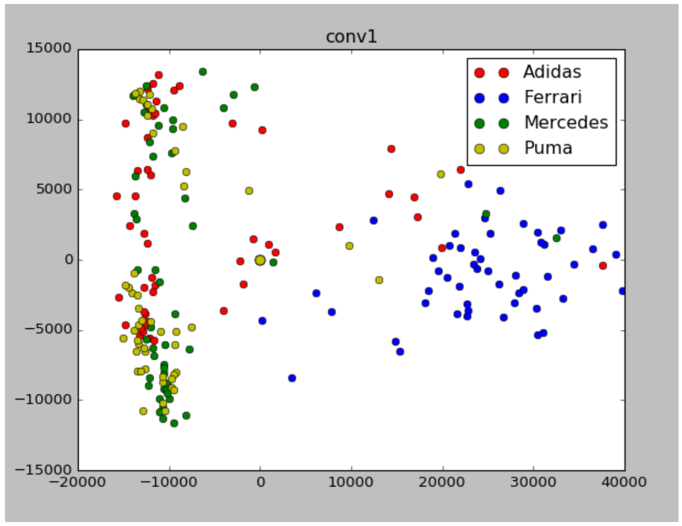

Figure 4.1. Spatial Arrangement of Activations at CONV1.

The scatter plot of Figure 4.1 represents the spatial arrangement and discriminability of the activations extracted from the convolutional layer CONV1 of the CNN pretrained on ImageNet.

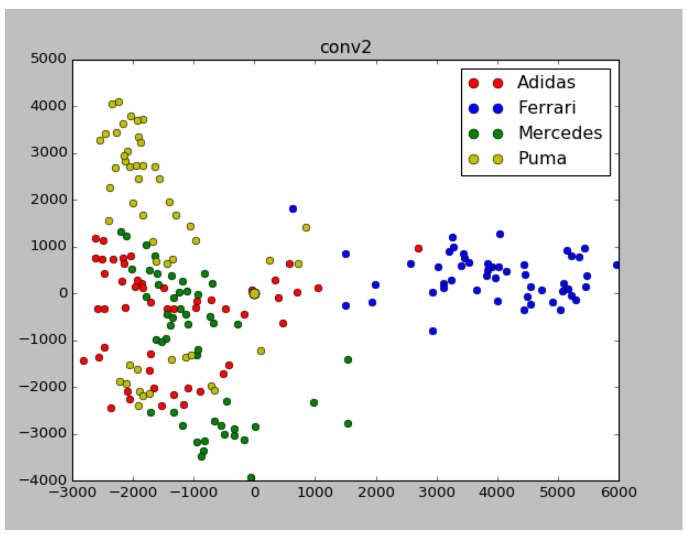

Figure 4.2. Spatial Arrangement of Activations at CONV2.

The scatter plot of Figure 4.2 represents the spatial arrangement and discriminability of the activations extracted from the convolutional layer CONV2 of the CNN pretrained on ImageNet.

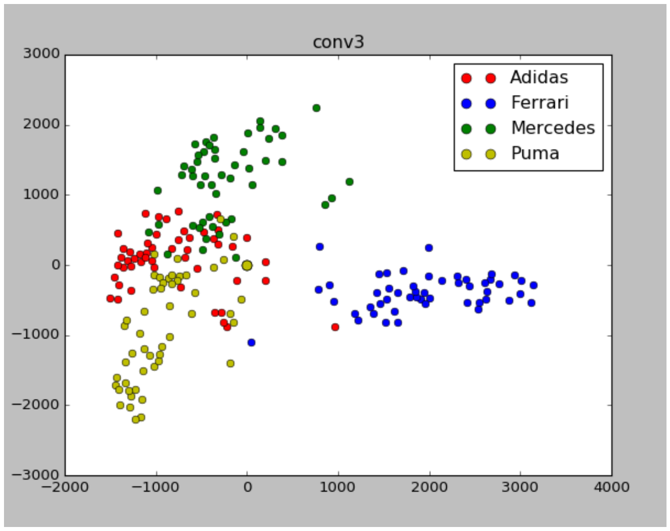

Figure 4.3. Spatial Arrangement of Activations at CONV3.

The scatter plot of Figure 4.3 represents the spatial arrangement and discriminability of the activations extracted from the convolutional layer CONV3 of the CNN pretrained on ImageNet.

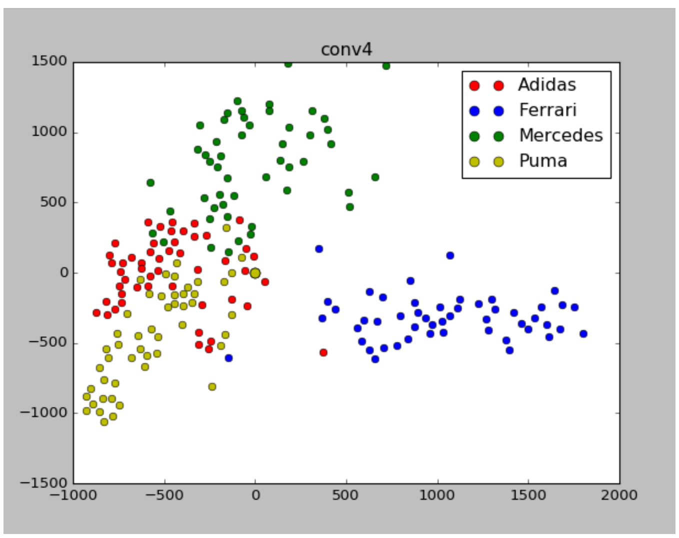

Figure 4.4. Spatial Arrangement of Activations at CONV4.

The scatter plot of Figure 4.4 represents the spatial arrangement and discriminability of the activations extracted from the convolutional layer CONV4 of the CNN pretrained on ImageNet.

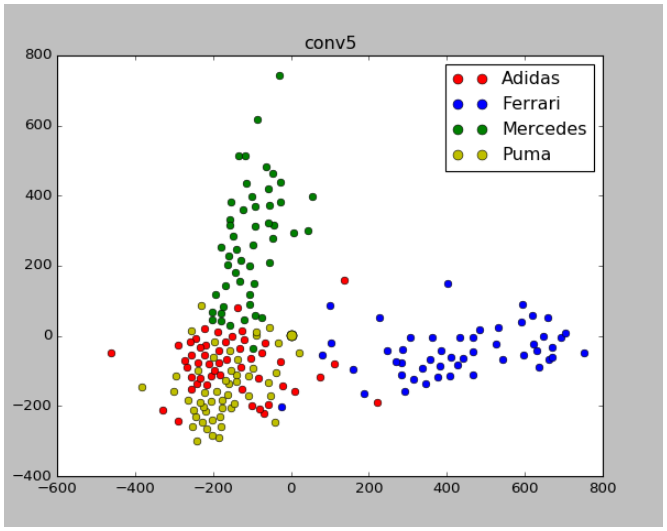

Figure 4.5. Spatial Arrangement of Activations at CONV5.

The scatter plot of Figure 4.5 represents the spatial arrangement and discriminability of the activations extracted from the convolutional layer CONV5 of the CNN pretrained on ImageNet.

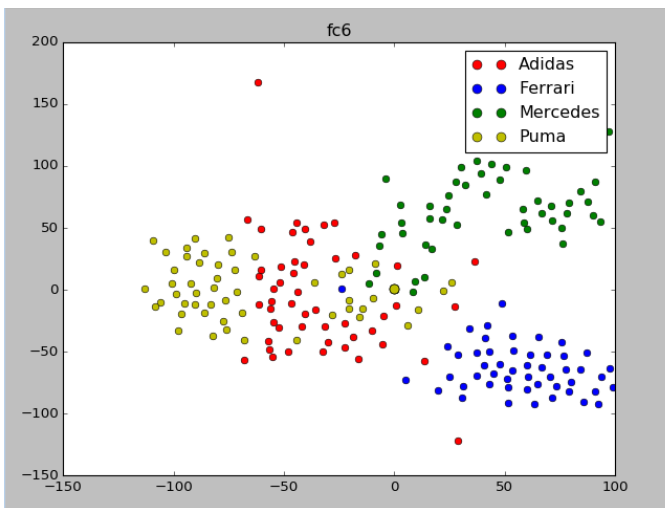

Figure 4.6. Spatial Arrangement of Activations at FC6.

The scatter plot of Figure 4.6 represents the spatial arrangement and discriminability of the activations extracted from the fully connected layer FC6 of the CNN pretrained on ImageNet.

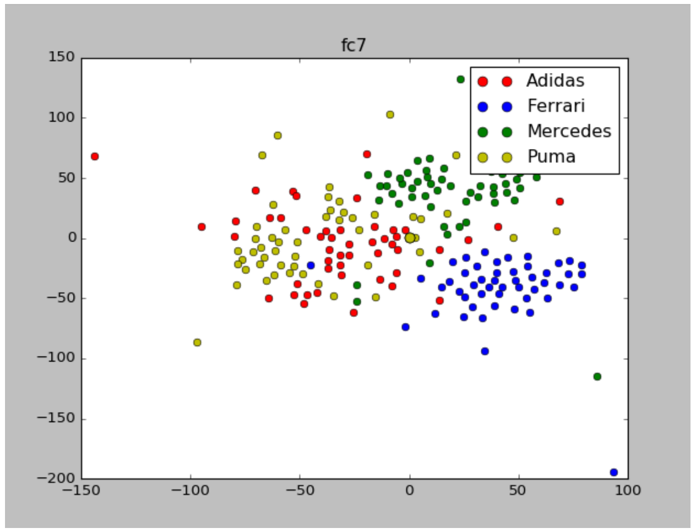

Figure 4.7. Spatial Arrangement of Activations at FC7.

The scatter plot of Figure 4.7 represents the spatial arrangement and discriminability of the activations extracted from the fully connected layer FC7 of the CNN pretrained on ImageNet.

Finding 4.1. *From visual observation of scatter plots created with the t-SNE technique, the fully connected layers FC6 and FC7 better represent the features' compactness in high dimensional space.*

In the following subsections, we investigate the appropriate parameters including nearest neighbors k, distance metric, number of training samples, and layer to extract features from to be considered to achieve the best results from our classification approach. The dataset for the following experiments is the logos dataset [27], [28]. The datataset is split into a training set and one test sample extracted from one of the 19 classes. For each iteration, we pick a test sample and keep 49 positive training samples (from the same class) and 50*18 negative samples (from classes other than the target class). During our experiment, we increment the number of training samples to be considered from every class at each iteration. To assess the accuracy of our experiment results we are utilizing the F score (or F1 score) function defined by the following formula:

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.1}$$

where

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

$$Recall = \frac{\text{TP}}{\text{TP} + FalseNegative(\text{FN})} \tag{4.3}$$

4.3 Comparing Different Values of the Nearest-Neighbors Parameter k

In this section we investigate the impact of the number of nearest neighbors /k/ on the performance of the k-NN classification task. We consider the fully connected layer FC6, and we check the F-scores for both metrics, the CS and the ED metrics, while varying k. Figure 4.8 shows F-score plots from experiments at layer FC6 with ED as the metric, while Figure 4.9 presents F-score plots from experiments at the same layer but with CS as the metric. For both figures, the x-axis represents the number of training instances s picked from each class.

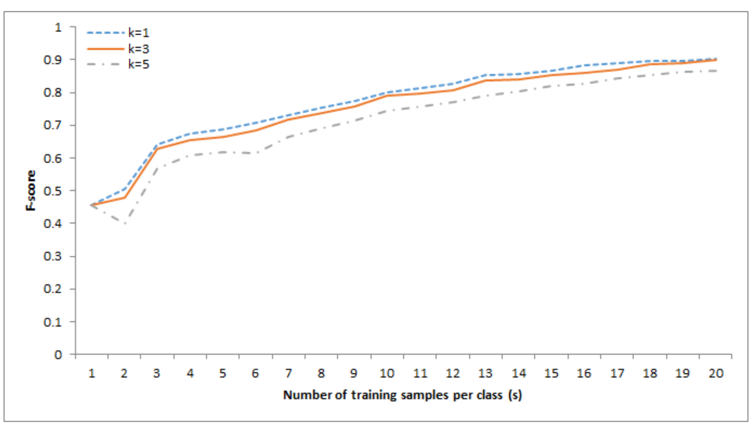

Figure 4.8. F-score for Different Values of k (layer=FC6, metric=ED, s=1:20).

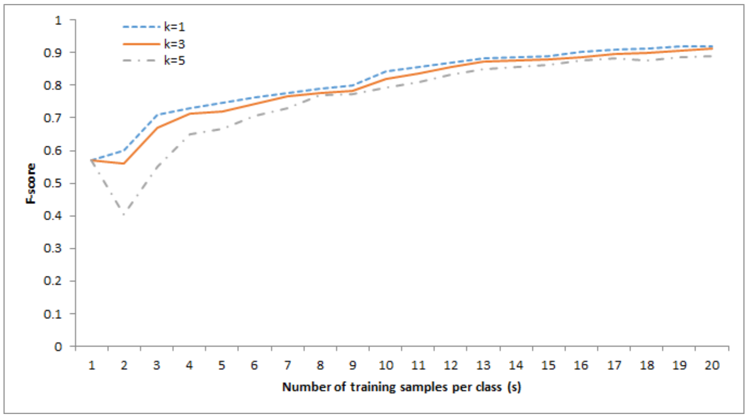

Figure 4.9. F-score for Different Values of k (layer=FC6, metric=CS, s=1:20).

From the previous results, k=1 achieves better classification results at the FC6 layer for both metrics, CS and ED. We replicate the same experiments at the layer FC7. We plot the F-score plots for a different number of nearest neighbors k. Figure 4.10 shows F-score plots from experiments at layer FC7 with ED as the metric, while Figure 4.11 presents F-score plots from experiments at the same layer but with CS as the metric. The results at FC7 support that k=1 is best choice for the number of nearest neighbors as found at FC6. Note that 1-NN in general leads to overfitting since the estimation of the class is based on the closest neighbor, which could be an outlier, mislabeled sample, or noise; yet, this not the case for our logos dataset. In addition, 1-NN classification works very well with a dataset with great separation between target values. Thus, we conclude that overfitting is not an issue for the logo dataset. For the value of k, we limited our search to 1, 3, and 5 as there seems to be a trend, worse performance with bigger k.

Finding 4.2. *k-NN algorithm with k=1 gives better classification results compared to k=3 and k=5 for both metrics CS and ED at different layers FC6 and FC7 .*

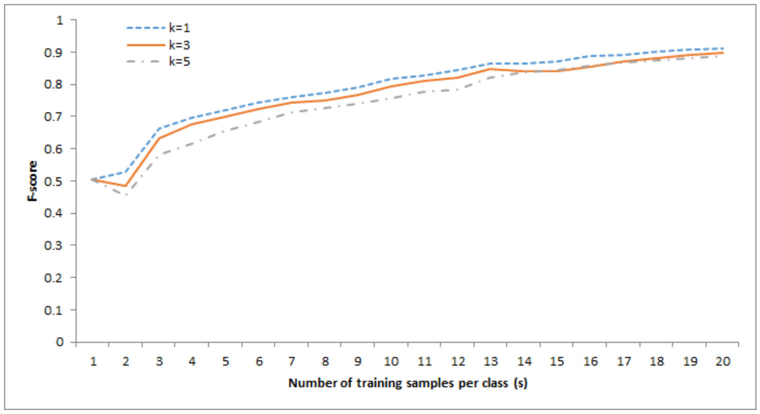

Figure 4.10. F-score for Different Values of k (layer=FC7, metric=ED, s=1:20).

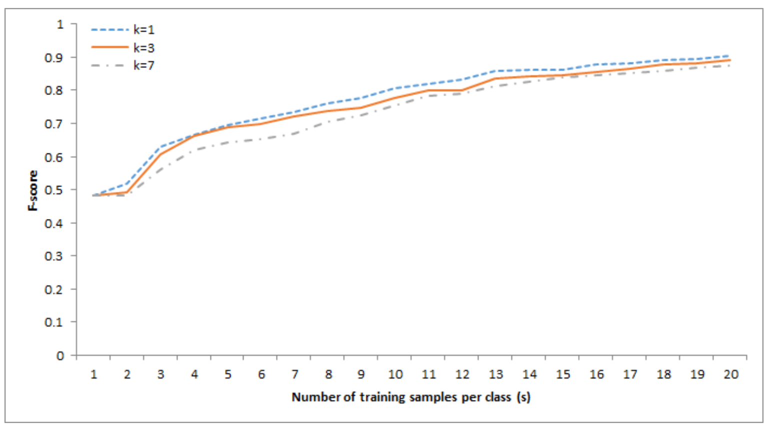

Figure 4.11. F-score for Different Values of k (layer=FC7, metric=CS, s=1:20).

4.4 Comparing the Two Metrics: CS and ED

In this section we investigate which metric, CS or ED, to be applied to features in order to find the training samples most similar to our target instance. Based on the previous finding, we will only consider k=1 for our *k*-NN approach in further experiments. Starting with FC6, we vary the number of training instances per class *s* (x-axis), and we plot the F-score (Y-axis) for both metrics. We repeat the experiments for FC7. Figure 4.12 shows that the F-score resulting from the *k*-NNs method with k=1 applied to the features extracted from FC6 with the CS metric tops the F-score resulting from the same experiment but with ED as the metric. Our hypothesis is that the CS is more appropriate than the ED as a metric for

our classification method based on the k-NN algorithm with k=1.

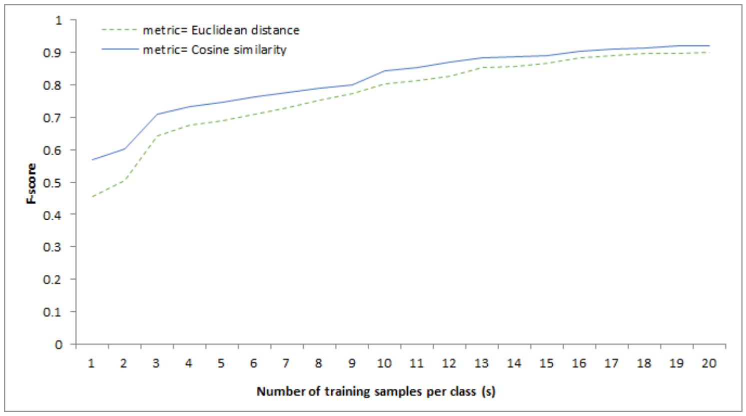

Figure 4.12. F-score for Different Metrics (layer=FC6, k=1, s=1:20).

To verify our hypothesis, we run further experiments at the layer FC7 with the k-NN algorithm for k=1. As shown in Figure 4.13, we plot the F-scores for the different metrics ED and CS in function of the number of training samples per class s (x-axis). Note that the F-scores presented by the Figure 4.12 and Figure 4.13 have very similar curves but we can conclude that the cosine similarity is more appropriate for the classification task.

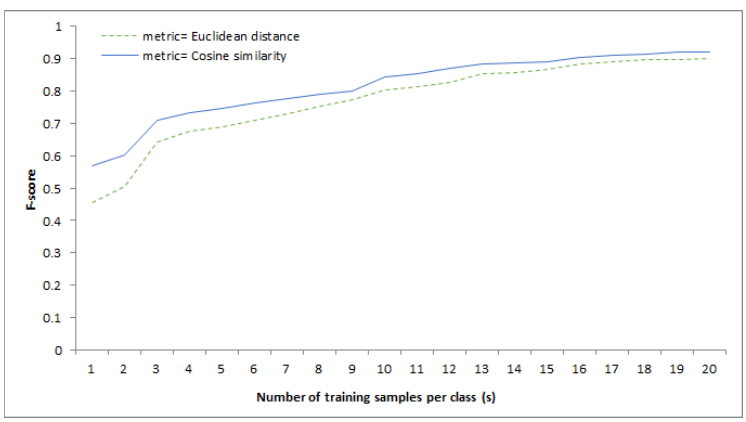

Figure 4.13. F-score for Different Metrics (layer=FC7, k=1, s=1:20).

Finding 4.3. *The CS as a metric leads to better classification than the ED metric with k-NN for k=1 at layers FC6 and FC7 .*

In the following section we investigate the best layer at which to extract our features in order to reach better classification results.

4.5 Comparing the Two Fully Connected Layers: FC6 and FC7

For the following discussion, we consider k=1 for the k-NN algorithm based on finding 4.2 and the CS metric based on finding 4.3, and we investigate from which layer to extract features. We plot the F-score at both layers FC6 and FC7. The x-axis presents the number of training instances s picked from each class. Figure 4.14 shows that the results of experiments based on the features extracted from FC6 outperform the ones derived from FC7 while applying the k-NN with k=1 and the CS metric as the parameter. Note that the curves are the same as the CS (top) curves in Figure 4.12 and Figure 4.13.

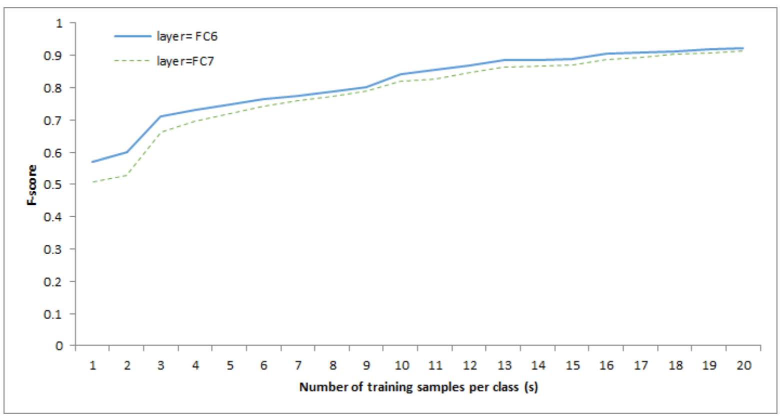

Figure 4.14. F-score for Different Layers (k=1, metric=CS, s=1:20).

Finding 4.4. *The fully connected layer FC6 is the best layer to extract the features from (k=1 and metric=CS).*

In the following section we investigate how many training samples per class should be considered in order to reach a specific classification performance.

4.6 Comparing Different Numbers of Training Samples s per Class

In this section we investigate the impact of the number of training samples s to be extracted from each class on the performance of a k-NN classification task. Based on the previous findings, we will only consider k=1, the fully connected layers FC6, and the CS metric for further experiments. We vary the number of training instances s picked from each class, and we record the F-score for each value of s. The results we are presenting in this section are as expected: better F-scores are obtained when a greater number of training samples per class is available. Moreover, the shown results are rather reliable because they are the average for over a dozen objects (19 classes).

Figure 4.15 shows that the F-score for the particular parameters of our experiments (k=1, metric=CS and layer= FC6) increases as the number of training samples per class s increases.

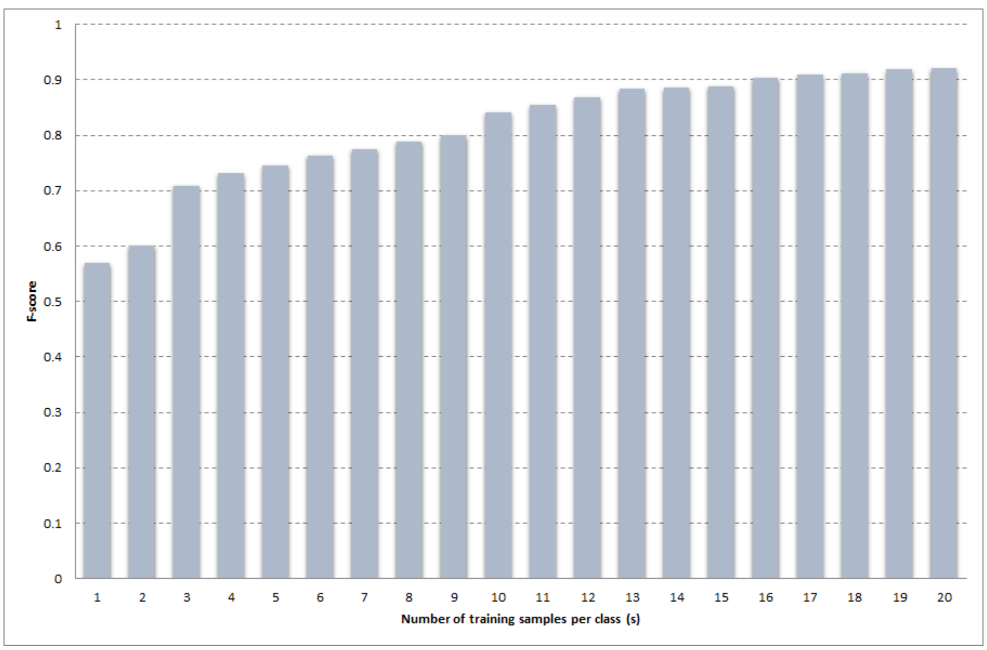

Figure 4.15. F-score for Different Values of Training Samples per Class.

Finding 4.5. *For almost any number of training samples per class* s_i *greater than* s_j*, we achieve better classification results.*

As a summary of our findings, we can state that the k-NN algorithm with the choice of parameters k=1, metric= CS, and layer=FC6 achieves better classification results. These results improve as we increase the number of training samples per class s.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: Conclusion

There are many challenges that arise in Computer Vision, and the classification of images is one of the core challenges. Computer Vision researchers have been using CNNs increasingly as they become a more viable approach for image classification. Unfortunately, an excessive number of training samples is needed when it comes to training CNNs, and this task is costly and time consuming. In addition, we can often not find sufficient training data for a specific class target; thus, it is best to reuse the knowledge that has been obtained from a previous model trained on a large dataset through the transfer learning approach.

Throughout our thesis work, new representations of the images were generated as we conducted a transfer learning of a deep CNN model. We started with a CNN trained on a huge dataset [9]. The weights of this model were transferred to our model. In our approach, we freeze these weights and feed training images to our model in order to generate new presentations of these inputs called features. Our training dataset consisted of logo images cropped from an original logo dataset publicly released [27], [28].

As we transferred the knowledge from a deep CNN model trained on a huge dataset, time to build the model from start to finish has been avoided and the expense of data labeling has been reduced. The features obtained from our scarce training data were within a high dimensional space (e.g., 4096, if extracted from the sixth convolutional layer (CONV6)). These features were the inputs to the k-NN algorithm in order to accomplish a classification of out-of-sample images. Through our methodology, we sought the most appropriate configuration to best handles the scarcity of labeled training data. We experimented with multiple parameters, including the distance metric, the layer from which the features were extracted, the number of nearest neighbors k to vote, and the available number of training samples, to achieve the greatest results in classification.

In order to investigate the best layer of the CNN to extract features from, we started with observing the dispersion and discrimination of the features extracted from the various layers of the CNN by projecting them onto a 2-D plane. The t-SNE [26] technique was chosen to visualize the high dimensional data, and the results show that the features in the first layers appeared to be less discriminative and more spread out. The features showed less dispersion towards the remaining last layers. Then, we focused our experiments on the last two layers, FC6 and FC7.

To determine the best value for the hyperparameter k, the number of nearest neighbors, and the best metric to be considered in high dimensional space, we ran our experiments for different values of k (1, 3 and 5), and at each iteration we considered two different metrics (Euclidean distance and cosine similarity). The accuracy of our classification was evaluated for each setting by collecting the number of True Positives (TPs), False Positives (FPs), and False Negatives (FNs) as the output the k-NN classifier. Since we were interested mainly in scarce data scenarios, we simulated small training datasets by choosing a specific number s of training instances from each class. This hyperparameter s was also varied in order to study its impact on the accuracy of the classification task.

Our final results show that the best output layer at which to extract the CNN activations is the FC6 layer. These results can be explained by the fact that the activations extracted from the layer FC6 are not general like those from the early layers. Furthermore, they are not specific to the target task, unlike FC7 or any classifier used in other research.

As for the metric, we compared the Euclidean distance to the cosine similarity in determining similar representations of images in high dimensional spaces. The cosine similarity was able to give better results in combination with the k-NN algorithm we used in our research, and it is more likely to be used in such a space dimension.

In regards to the effect of the size of the training set, we found that we can reach good results by using the k-NN solution applied to the activation codes extracted from the FC6 layer for small datasets. For instance, we achieved 90% accuracy for only 17 training samples per class (19 classes were considered for our research), and the results improved as we increased the number of the training samples (in our case we varied s from 1 to 20).

Our work supports the mounting evidence that the activations extracted from CNNs are very efficient. Unlike previous researches we methodically evaluate the impact of the number of training samples on classificaiton performance while applying a nonparametric method, the k-NN classifier. Furthermore, our research helps building a detector for objects similar to logos with few training samples for a specific level of accuracy. For example, with 17 training samples we expect a 90% classification accuracy.

Practically, we encourage using the solution that has been proposed in this thesis to implement sophisticated classification systems that overcome the shortage of training images and support tactical decision making. Our work has determined the configuration that best tolerates the lack of data while still being able to offer a desired level of classification accuracy, lower costs, and less computational time.

For future work, we suggest further research and investigation of the fine-tuning as a transfer learning scenario, instead of freezing the transferred layers, and applying our proposed parameters including number of nearest neighbors k, the metric, and the number of training samples. The results from this can be compared with the results we have obtained.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Main Algorithm

Input:

- className : dictionary of class names
- m_nic: maximum number of training instances per class

Variables:

- kn: number of nearest neighbors
- target: name of the target class
- testInstance: name of the test instance
- dicTrain: dictionary of all train instances
- trainSet: train instance for current 'testInstance'
- l: index of the 'testInstance' within a target class
- s: number of samples to be considered from each class
- eLabels: labels of 'kn' nearset neighbors for Euclidean distance metric
- eVote: the predicted class for a 'testInstance' for Euclidean distance metric
- cLabels: labels of 'kn' nearset neighbors for cosine similarity metric
- cVote: the predicted class for a 'testInstance' cosine similarity metric

Output:

- eTP: dictionary having predicted class, kn and s as keys and number of true positives as value for Euclidean distance metric
- eFP: dictionary having predicted class, kn and s as keys and number of false positives as value for Euclidean distance metric
- cTP: dictionary having predicted class, kn and s as keys and number of true positives as value for cosine similarity metric
- cFP: dictionary having predicted class, kn and s as keys and number of false positives as value for cosine similarity metric

```
for varName in className do
  target ← varName
  for kn in {1,3,5} do
    for l in  instances of target class  do
```

```
            testInstance ← instance l
            dicTrain ← all_data_except_ instance_ l
            for s ← 1 to m_nic do
               trainSet ← s instances per class from dicTrain
               # k nearest neighbors with Euclidean distance
               tree ← call KDTree ( training_set = trainSet and metric = Euclidean distance )
               eLabels ← query the tree ( test_set = testInstance and k = kn )
               # k nearest neighbors with cosine similarity
               cLabels ← call ckNeighbors(training_set=trainSet, test_set=testInstance,k=kn)
               eVote ← call classPrediction(eLabels)
               cVote ← call classPrediction(cLabels)
               if eVote = target then
                  increment value of eTP( class= eVote, k=kn , number-of-sample-per-class =s)
               else
                  increment value of eFP( class= eVote, k=kn , number-of-sample-per-class =s)
               end if
               if cVote = target then
                  increment value of cTP( class= cVote, k=kn , number-of-sample-per-class =s)
               else
                  increment value of cFP( class= cVote, k=kn , number-of-sample-per-class =s)
               end if
            end for
         end for
      end for
   end for
```

APPENDIX B:
Numerical Results

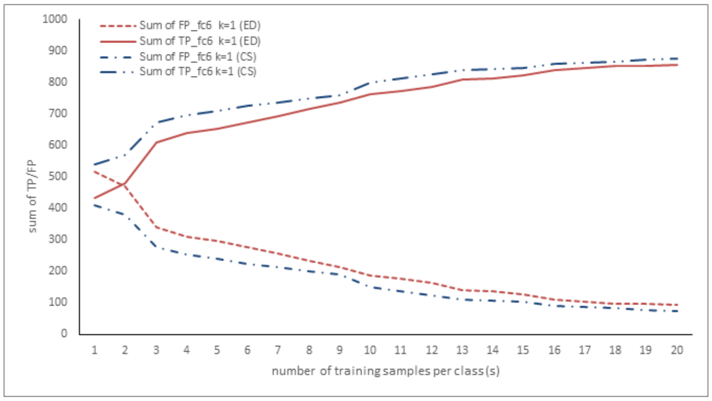

Figure B.1. Variation of TP and FP for Different Metrics (fc6, k=1).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=1 applied to the features extracted from the sixth layer (fc6) for different metrics: cosine similarity (CS) and Euclidean distance (ED).

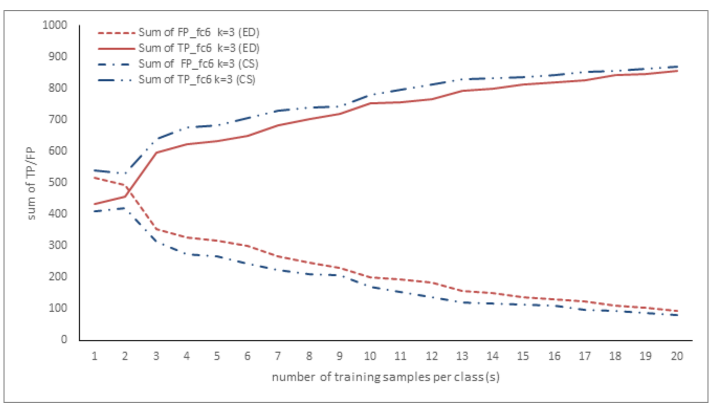

Figure B.2. Variation of TP and FP for Different Metrics (fc6, k=3).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=3 applied to the features extracted from the sixth layer (fc6) for different metrics: cosine similarity (CS) and Euclidean distance (ED).

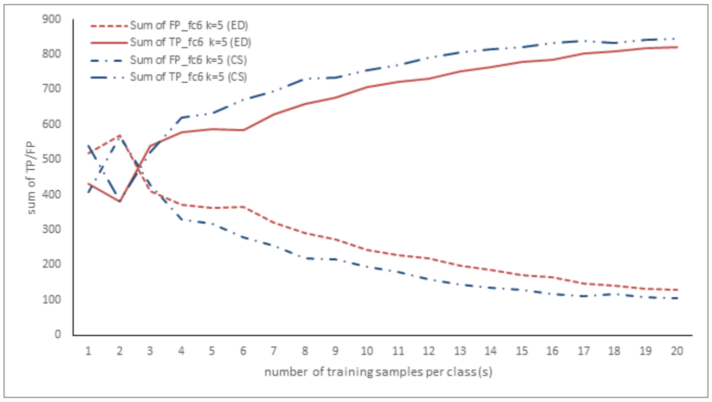

Figure B.3. Variation of TP and FP for Different Metrics (fc6, k=5).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=5 applied to the features extracted from the sixth layer (fc6) for different metrics: cosine similarity (CS) and Euclidean distance (ED).

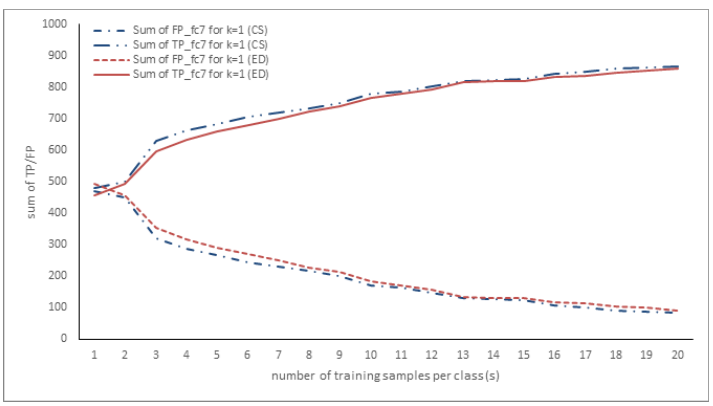

Figure B.4. Variation of TP and FP for Different Metrics (fc7, k=1).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=1 applied to the features extracted from the seven layer (fc7) for different metrics: cosine similarity (CS) and Euclidean distance (ED).

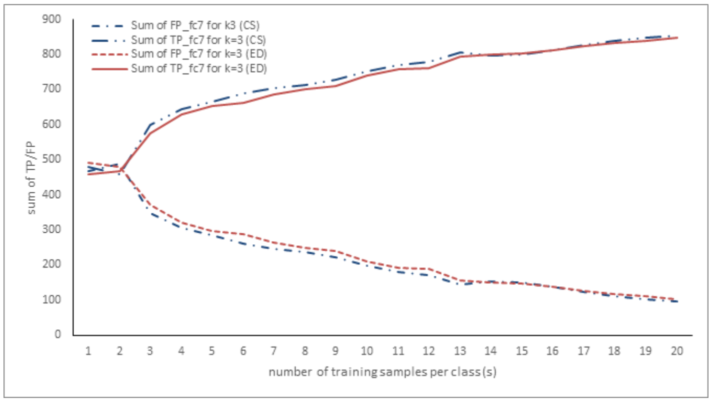

Figure B.5. Variation of TP and FP for Different Metrics (fc7, k=3).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=3 applied to the features extracted from the seven layer (fc7) for different metrics: cosine similarity (CS) and Euclidean distance (ED). .

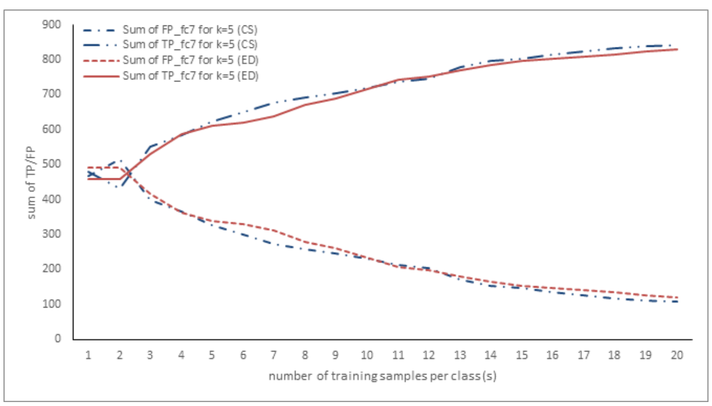

Figure B.6. Variation of TP and FP for Different Metrics (fc7, k=5).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=5 applied to the features extracted from the seventh layer (fc7) for different metrics: cosine similarity (CS) and Euclidean distance (ED).

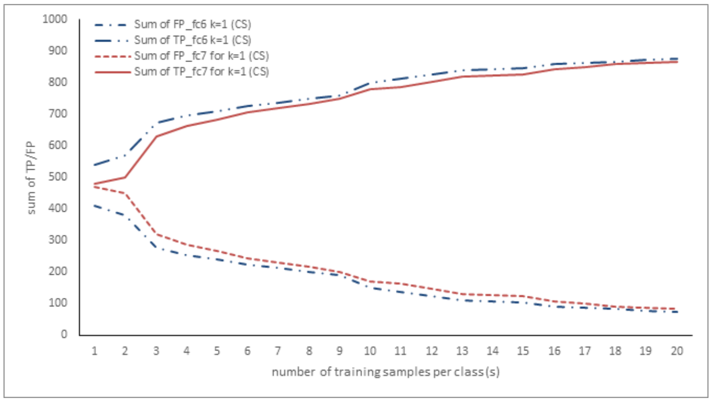

Figure B.7. Variation of TP and FP for Different Layers (CS, k=1).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=1 and metric= cosine similarity (CS) applied to the features extracted from the seventh layer (fc7) and the sixth layer (fc6).

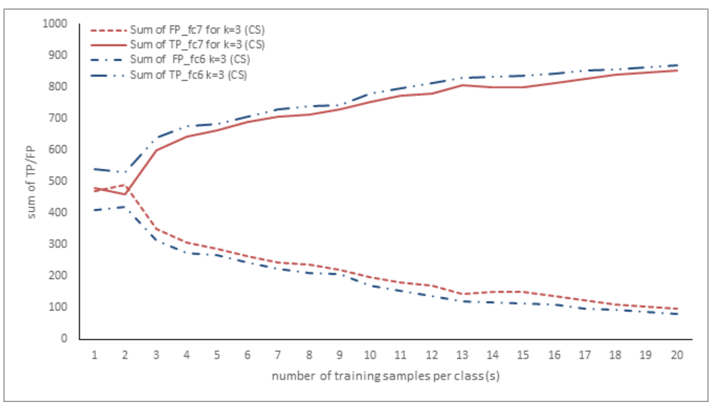

Figure B.8. Variation of TP and FP for Different Layers (CS, k=3).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=3 and metric= cosine similarity (CS) applied to the features extracted from the seventh layer (fc7) and the sixth layer (fc6).

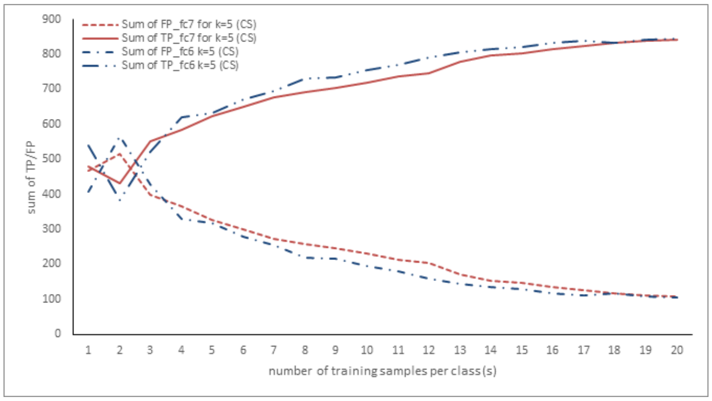

Figure B.9. Variation of TP and FP for Different Layers (CS, k=5).

Variation of sums of true positives (TP) and false positives (FP) as a result of the kNN method with k=5 and metric= cosine similarity (CS) applied to the features extracted from the seventh layer (fc7) and the sixth layer (fc6).

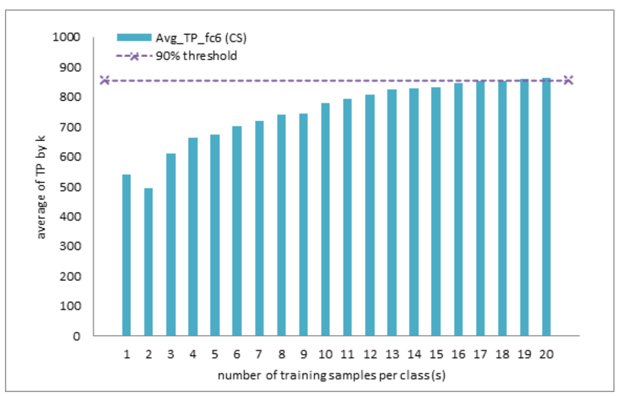

Figure B.10. Average of TP for Different Values of k (fc6, metric=CS).

Positive correlation between the number of training instances s (x-axis) picked from each class and the average of true positives (TP) for different values of k. The cosine similarity (CS) metric is applied to features extracted from the sixth layer (fc6). A 90% accuracy in terms of TPs is achieved for s equal or greater than 17 instances per class.

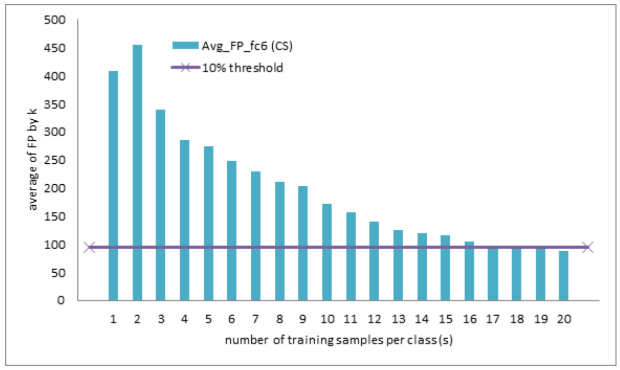

Figure B.11. Average of FP for Different Values of k (fc6, metric=CS).

Negative correlation between the number of training instances s (x-axis) picked from each class and the average of false positives (FP) for different values of "k". The cosine similarity (CS) metric is applied to features extracted from the sixth layer (fc6). For s equal or greater than 17, we have less than 10% inaccuracy in terms of FPs

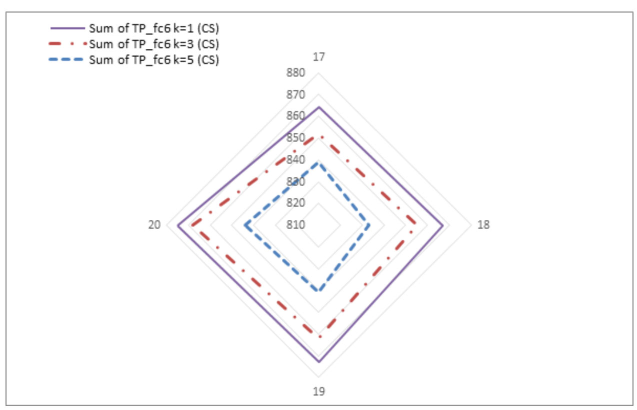

Figure B.12. Sum of TP for Different Values of k (fc6, metric=CS, s=17:20).

Positive correlation between the number of training instances "s" (x-axis) picked from each class and the number of true positives (TP).

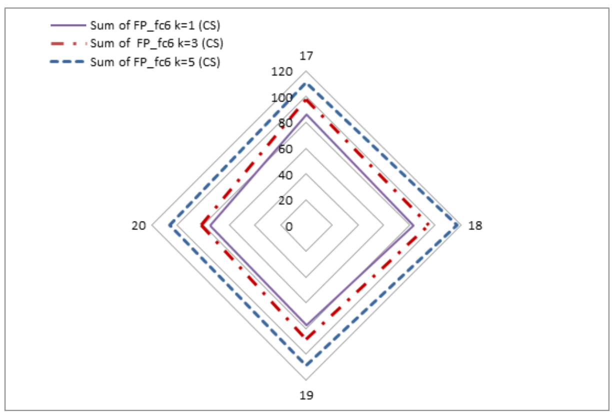

Figure B.13. Sum of FP for Different Values of k (fc6, metric=CS, s=17:20).

Negative correlation between the number of training instances "s" (x-axis) picked from each class and the number of false positives (FP).

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: Results by Logo Classes

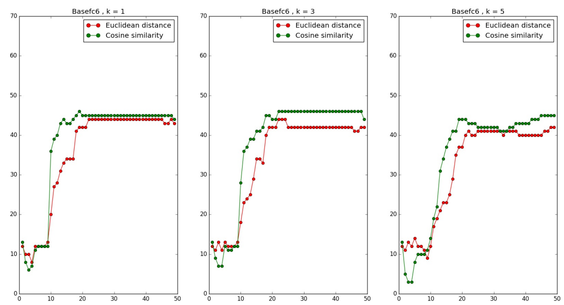

Figure C.1. TP for the Base Logo (layer=FC6).

The The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Base class for k=1, k=3, and k=5.

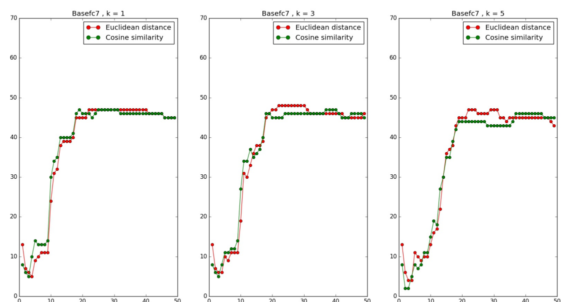

Figure C.2. TP for the Base Logo (layer=FC7).

The The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Base class for k=1, k=3, and k=5.

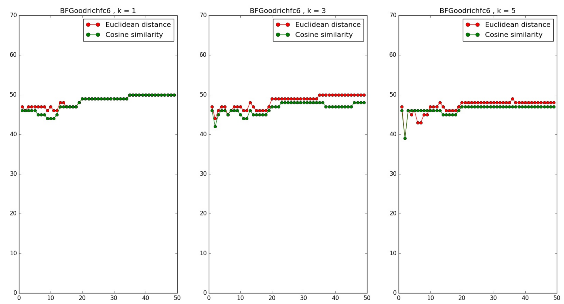

Figure C.3. TP for the BFGoodrich Logo (layer=FC6).

The x-axis represents the number of samples s per class while the y-axis represents the number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the BFGoodrich class for k=1, k=3, and k=5.

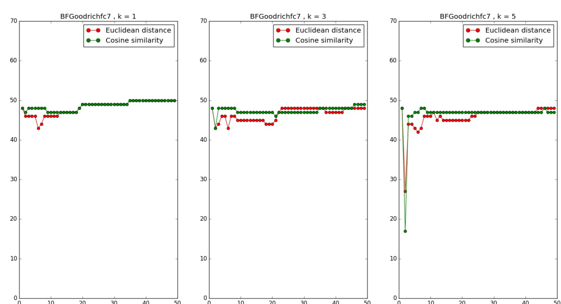

Figure C.4. TP for the BFGoodrich Logo (layer=FC7).

The x-axis represents the number of samples s per class while the y-axis represents the number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the BFGoodrich class for k=1, k=3, and k=5.

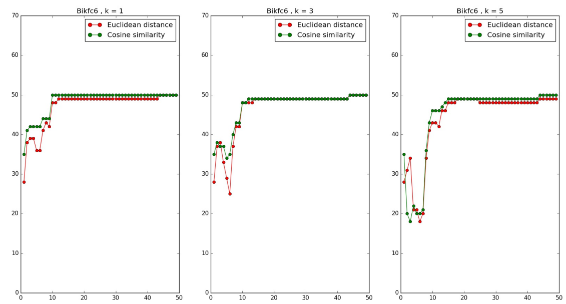

Figure C.5. TP for the Bik Logo (layer=FC6).

The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Bik class for k=1, k=3, and k=5.

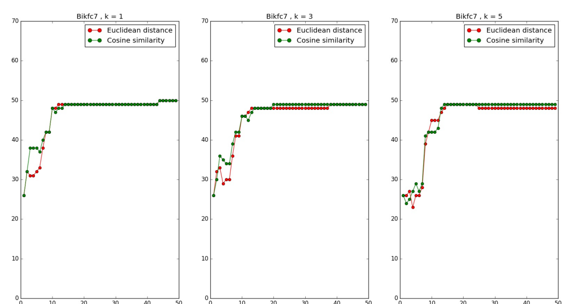

Figure C.6. TP for the Bik Logo (layer=FC7).

The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Bik class for k=1, k=3, and k=5.

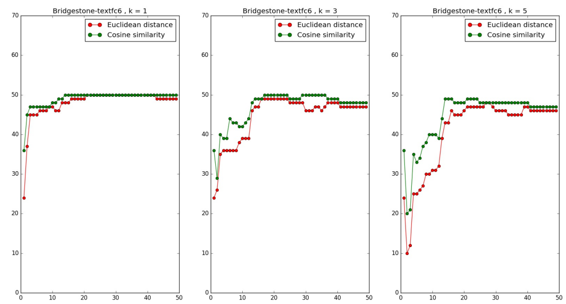

Figure C.7. TP for the Bridgestone-text Logo (layer=FC6).

The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Bridgestone-text class for k=1, k=3, and k=5.

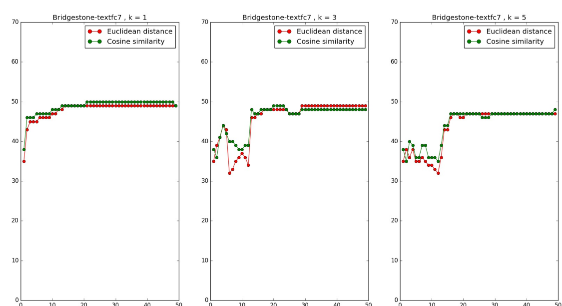

Figure C.8. TP for the Bridgestone-text Logo (layer=FC7).

The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Bridgestone-text class for k=1, k=3, and k=5.

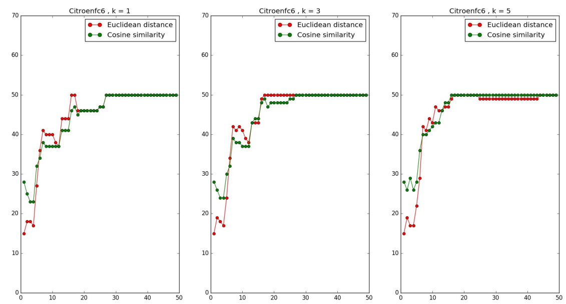

Figure C.9. TP for the Citroen Logo (layer=FC6).

The The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Citroen class for k=1, k=3, and k=5.

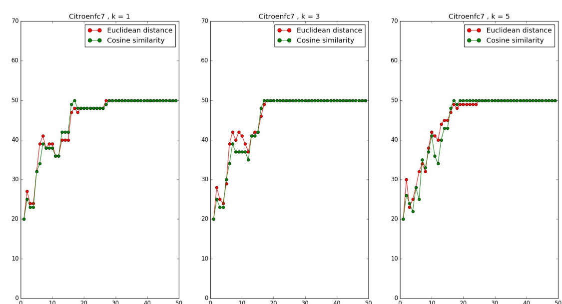

Figure C.10. TP for the Citroen Logo (layer=FC7).

The The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Citroen class for k=1, k=3, and k=5.

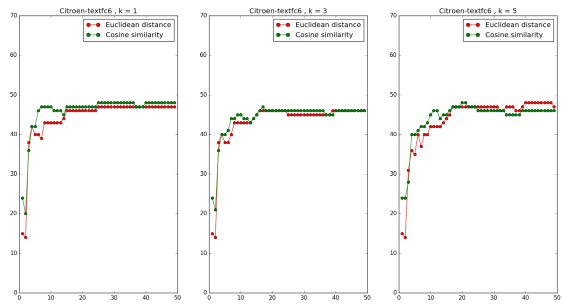

Figure C.11. TP for the Citroen-text Logo (layer=FC6).

The The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Citroen-text class for k=1, k=3, and k=5.

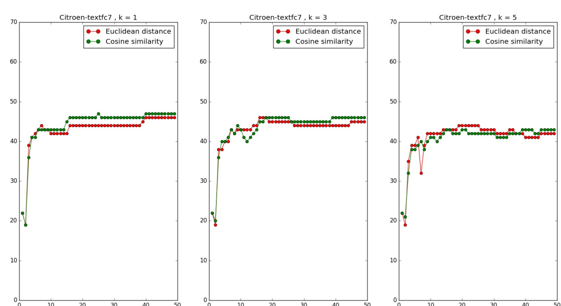

Figure C.12. TP for the Citroen-text Logo (layer=FC7).

The The x-axis represents the number of samples s per class while the y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Citroen-text class for k=1, k=3, and k=5.

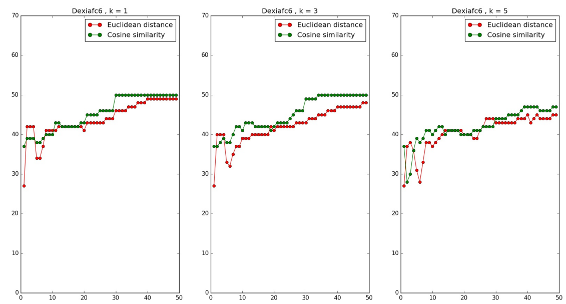

Figure C.13. TP for the Dexia Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Dexia class for k=1, k=3, and k=5.

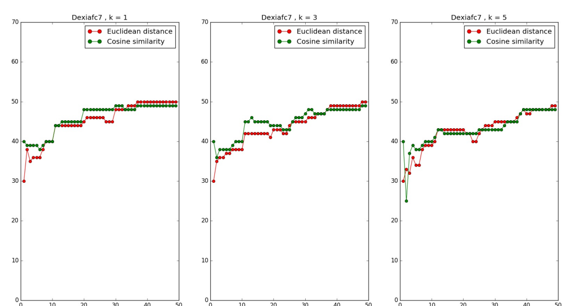

Figure C.14. TP for the Dexia Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Dexia class for k=1, k=3, and k=5.

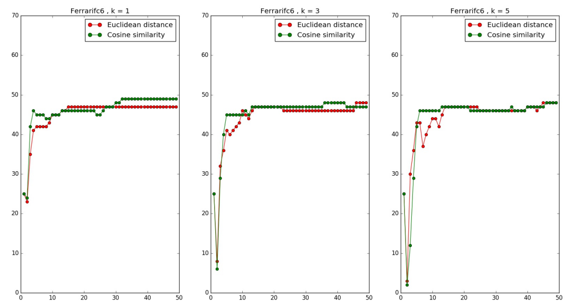

Figure C.15. TP for the Ferrari Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Ferrari class for k=1, k=3, and k=5.

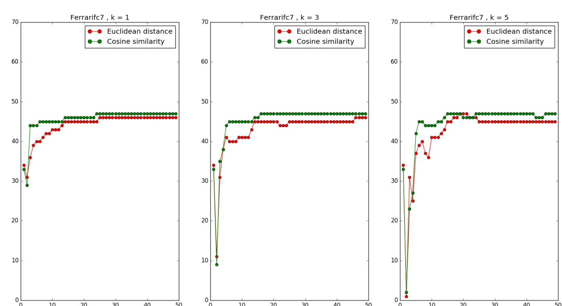

Figure C.16. TP for the Ferrari Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Ferrari class for k=1, k=3, and k=5.

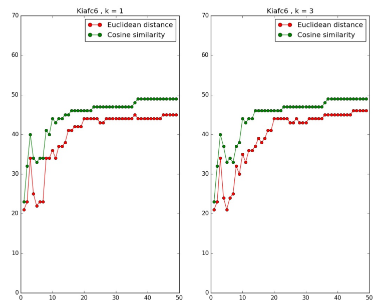

Figure C.17. TP for the Kia Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Kia class for k=1, k=3, and k=5.

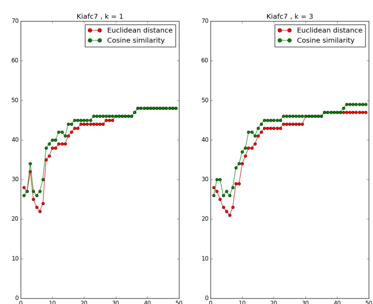

Figure C.18. TP for the Kia Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Kia class for k=1, k=3, and k=5.

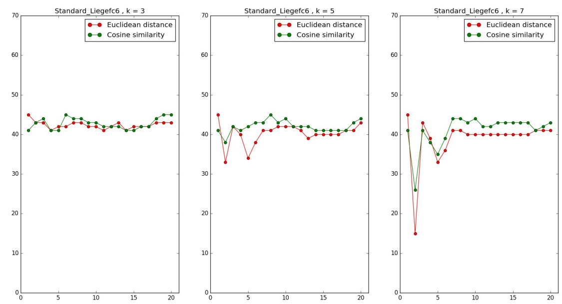

Figure C.19. TP for the Liege Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Liege class for k=1, k=3, and k=5.

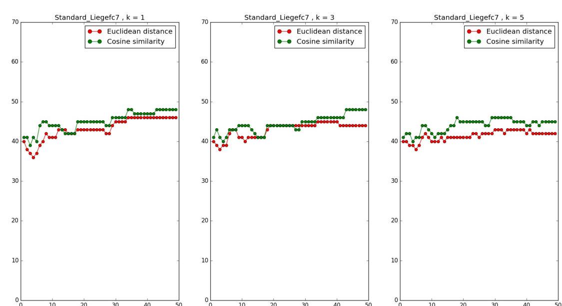

Figure C.20. TP for the Liege Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Liege class for k=1, k=3, and k=5.

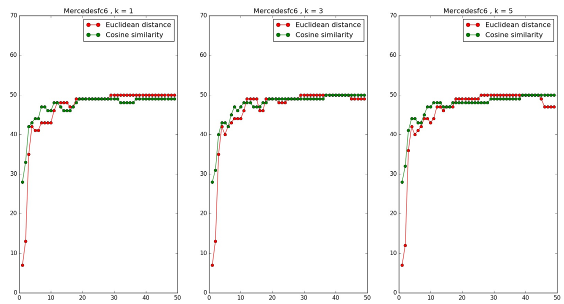

Figure C.21. TP for the Mercedes Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Mercedes class for k=1, k=3, and k=5.

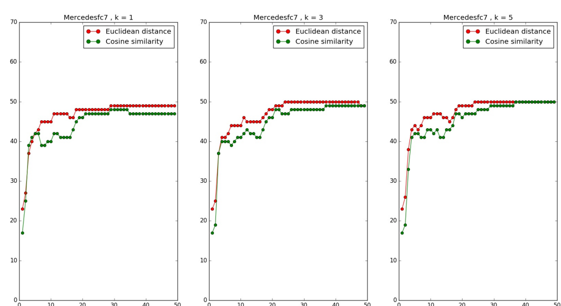

Figure C.22. TP for the Mercedes Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Mercedes class for k=1, k=3, and k=5.

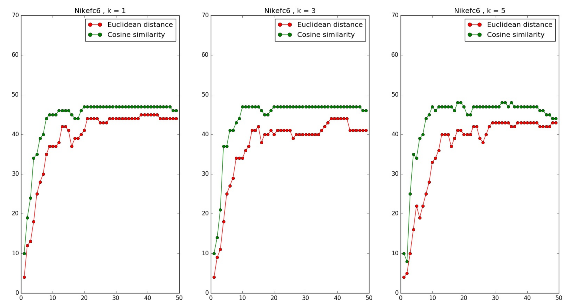

Figure C.23. TP for the Nike Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Nike class for k=1, k=3, and k=5.

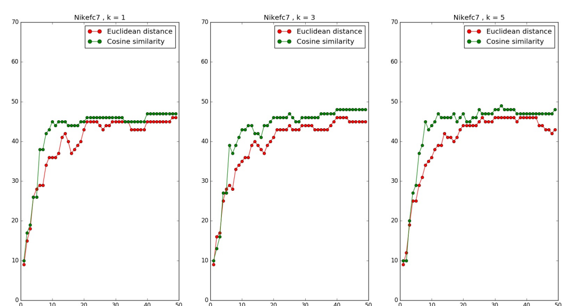

Figure C.24. TP for the Nike Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Nike class for k=1, k=3, and k=5.

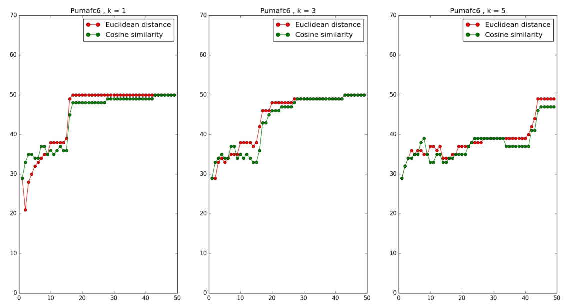

Figure C.25. TP for the Puma Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Puma class for k=1, k=3, and k=5.

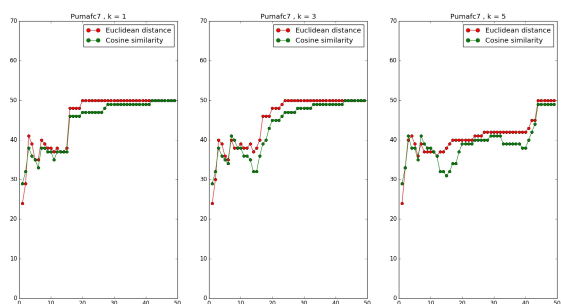

Figure C.26. TP for the Puma Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Puma class for k=1, k=3, and k=5.

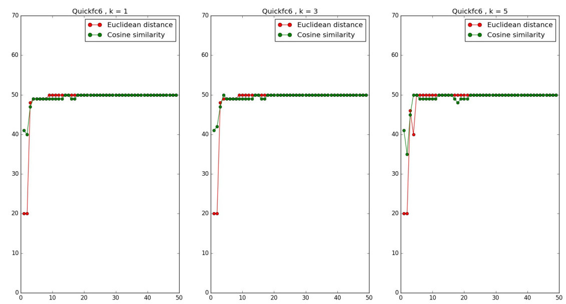

Figure C.27. TP for the Quick Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Quick class for k=1, k=3, and k=5.

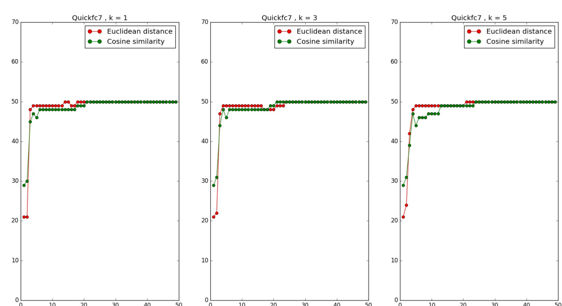

Figure C.28. TP for the Quick Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Quick class for k=1, k=3, and k=5.

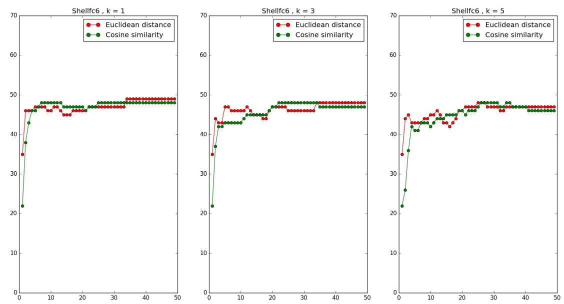

Figure C.29. TP for the Shell Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Shell class for k=1, k=3, and k=5.

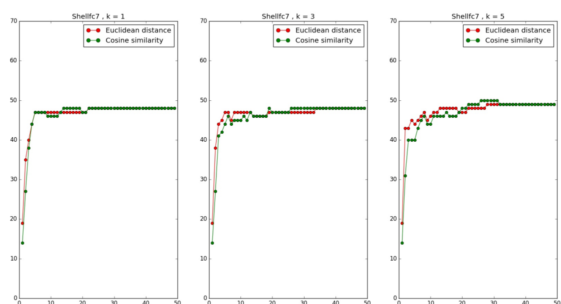

Figure C.30. TP for the Shell Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Shell class for k=1, k=3, and k=5.

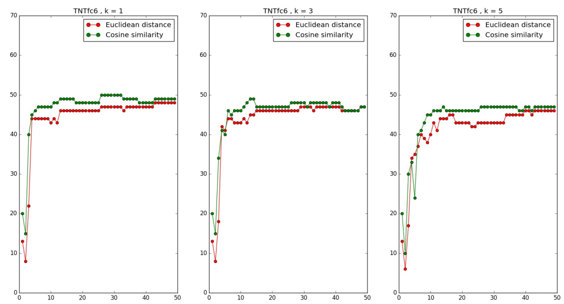

Figure C.31. TP for the TNT Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the TNT class for k=1, k=3 and k=5.

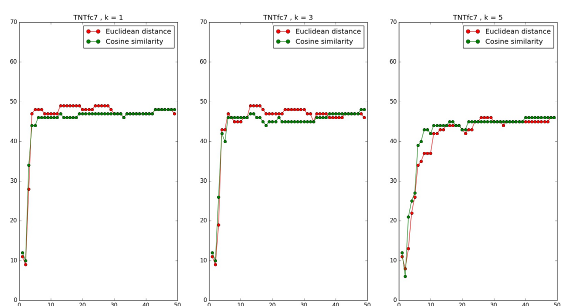

Figure C.32. TP for the TNT Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the TNT class for k=1, k=3, and k=5.

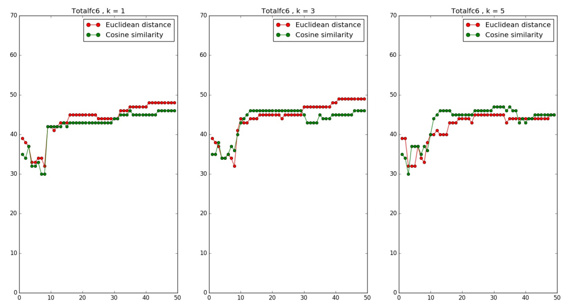

Figure C.33. TP for the Total Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Total class for k=1, k=3, and k=5.

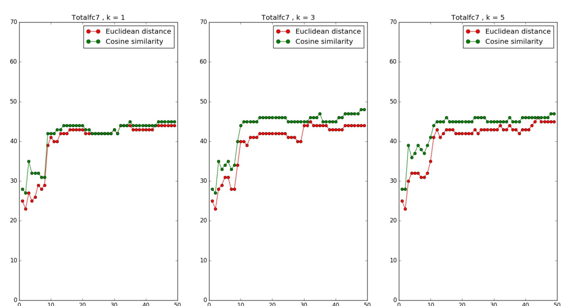

Figure C.34. TP for the Total Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Total class for k=1, k=3, and k=5.

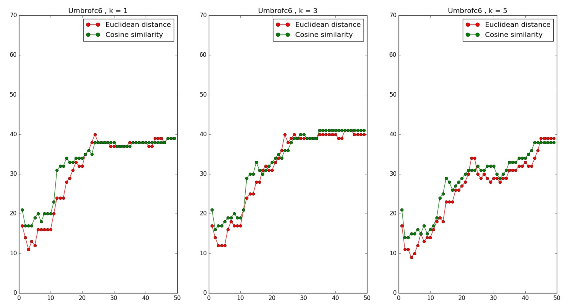

Figure C.35. TP for the Umbro Logo (layer=FC6).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC6. From left to right, these graphs represent results for test samples from the Umbro class for k=1, k=3, and k=5.

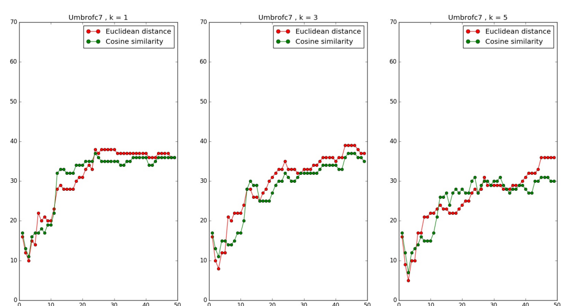

Figure C.36. TP for the Umbro Logo (layer=FC7).

The x-axis represents the number of samples s per class while y-axis represents number of TPs out of 50 attempts of the classification task at layer FC7. From left to right, these graphs represent results for test samples from the Umbro class for k=1, k=3, and k=5.

List of References

[1] D. G. Lowe, “Distinctive image features from scale invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *2001 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.

[3] P. Y. Simard, D. Steinkraus, and J. C. Platt, “Best practices for convolutional neural networks applied to visual document analysis,” in *ICDAR*, 2003, vol. 3, pp. 958–962.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[5] F.-F. Li and A. Karpathy. (2015). Convolutional Neural Networks for Visual Recognition. [Online]. Available: http://cs231n.github.io/convolutional-networks

[6] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, “ICA with reconstruction cost for efficient overcomplete feature learning,” in *Advances in Neural Information Processing Systems*, 2011, pp. 1017–1025.

[7] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 248–255.

[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[11] Y. LeCun, “Generalization and network design strategies,” *Connectionism in Perspective*, pp. 143–155, 1989.

[12] I. G. Y. Bengio and A. Courville, “Deep learning,” 2016, book in preparation for MIT Press. Available: http://www.deeplearningbook.org

[13] Y. Li, L. Liu, C. Shen, and A. van den Hengel, “Mid-level deep pattern mining,” in *2015 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 971–980.

[14] P. Fischer, A. Dosovitskiy, and T. Brox, “Descriptor matching with convolutional neural networks: a comparison to sift,” *arXiv preprint arXiv:1405.5769*, 2014.

[15] R. Szeliski, *Computer Vision: Algorithms and Applications*. Berlin, Germany: Springer Science & Business Media, 2010.

[16] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[17] Y. Bengio, “Deep learning of representations for unsupervised and transfer learning,” *ICML Unsupervised and Transfer Learning*, vol. 27, pp. 17–36, 2012.

[18] Y. Bengio, F. Bastien, A. Bergeron, N. Boulanger-Lewandowski, T. M. Breuel, Y. Chherawala, M. Cisse, M. Côté, D. Erhan, J. Eustache *et al.*, “Deep learners benefit more from out-of-distribution examples,” in *AISTATS*, 2011, pp. 164–172.

[19] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[20] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” *arXiv preprint arXiv:1310.1531*, 2013.

[21] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: An astounding baseline for recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[22] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” *arXiv preprint arXiv:1312.6229*, 2013.

[23] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in *2014 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[25] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *2014 European Conference on Computer Vision*. Springer, 2014, pp. 818–833.

[26] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[27] A. Joly and O. Buisson, "Logo retrieval with a contrario visual query expansion," in *Proceedings of the 17th ACM International Conference on Multimedia*, 2009, pp. 581–584.

[28] P. Letessier, O. Buisson, and A. Joly, "Scalable mining of small visual objects," in *Proceedings of the 20th ACM International Conference on Multimedia*. ACM, 2012, pp. 599–608.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California